Micro Focus Visual COBOL 2.3J for Windows

Pro*COBOL による Oracle データベース 12.1c

アクセス動作検証 検証結果報告書

2016 年 04 月 22 日

マイクロフォーカス株式会社

第２版

1. 検証概要、目的及びテスト方法

1.1 検証概要

Oracle データベースクライアントは、COBOL アプリケーションから Oracle データベースにアクセスするためのプログラミングツールとして Pro*COBOL というプリコンパイラを搭載しています。本ツールを利用しますと、COBOL プログラマは SQL 文を埋め込み形式でプログラムソース中に記述できます。Pro*COBOL プリコンパイラは、COBOL プログラム中に埋め込まれた SQL 文を標準の Oracle ランタイム・ライブラリ・コールに変換します。この Pro*COBOL によるプリコンパイル処理は、Micro Focus の COBOL コンパイラを含めた各種 COBOL コンパイラで翻訳可能な COBOL プログラムを生成します。

Windows 版の Pro*COBOL は本稿執筆時点の段階にて Visual COBOL より 1 つ前の世代の Micro Focus COBOL 開発環境製品シリーズ Micro Focus Net Express5.1 をサポートするとマニュアルに明記しています。

[Oracle Database クイック・インストレーション・ガイド 12c リリース 1 (12.1) for Microsoft Windows x64 (64-Bit)]
https://docs.oracle.com/cd/E49329_01/install.121/e48235/toc.htm#BGBEEBAD
（2016 年 04 月 18 日リンク確認）

しかしながら、Pro*COBOL によってプリコンパイルされた COBOL プログラムは極めて標準的な COBOL 文法に準拠したものであり、特定のコンパイラバージョンに依存する要因は考えられません。更に、Visual COBOL は稼働保証環境として指定された製品 Net Express 5.1 の後継製品にあたるものであり、コンパイラ機能については上位互換を保証しています。このため Net Express 5.1 向けに展開されるプリコンパイル後の COBOL ソースはそのまま Visual COBOL でも稼働します。

本稿では、Pro*COBOL よりプリコンパイル生成された COBOL プログラムを Visual COBOL でコンパイルし、 Oracle データベースへアクセスできることを動作検証しました。また、Visual COBOL に装備された COBSQL 統合プロセッサを利用し、Pro*COBOL によるプリコンパイル、Visual COBOL によるコンパイルをワンステップで処理できることも検証しました。本検証は 2015 年 12 月と 2016 年 4 月で計 2 回検証を行っています。1 回目は Visual COBOL 2.3J に対して、2 回目には Visual COBOL 2.3J Update 1 に対して検証しています。

1.2 目的及びテスト方法

Micro Focus Visual COBOL は、Pro*COBOL が稼働保証対象とする Net Express 5.1 のコンパイラ機能に対して上位互換を持つ最新鋭の COBOL 言語開発・実行環境製品です。Oracle は Pro*COBOL 用のサンプルプログラムを提供しています。本検証ではまずそれを使って Pro*COBOL と Visual COBOL の連携により生成されたモジュールが正常に Oracle データベースへアクセスし、データ操作をできることを検証しました。

Visual COBOL はコンパイル処理を発行する前に Oracle, Sybase, Informix のプリコンパイラを内部的に呼び出し、ワンステップでプリコンパイル及びコンパイルを処理する COBSQL 統合プロセッサを備えます。Oracle へアクセスするのであれば、Pro*COBOL を内部的に利用します。本機能を利用すれば、プリコンパイル後の埋め込み SQL 文をライブラリコールに置き換えた COBOL ソースではなく、実際にプログラマがメンテナンスする埋め込み SQL 文が入ったままのロジカルなソースを使って管理できます。更に、Visual Studio 2012, Visual Studio 2013, Visual Studio 2015, Eclipse の IDE に COBOL 開発用に作りこんだ Visual COBOL のエディタやデバッガでその埋め込み SQL 文が入ったソースを扱うことができるため、開発作業の生産性を著しく高めることができます。本稿では Pro*COBOL の検証に使用した埋め込み SQL 文が入ったままの状態のサンプルプログラムを使って Visual Studio 2015 IDE 上で COBSQL を利用した開発ができることも検証確認しました。

2. 検証環境

【第 1 回検証】

- **Database サーバー**

  ソフトウェア

  - Windows Server 2012R2
  - Oracle Database 12c Release 1(12.1.0.2.0) for Microsoft Windows(x64)

  ハードウェア

機種：	n/a(public クラウドを利用)
CPU：	Intel Xeon CPU E5-2670 v2 2.50GHz
Memory：	3.75 Gbyte

- **Database クライアント**

  ソフトウェア

  - Windows 10 Enterprise
  - Oracle Database 12c Release 1 Client(12.1.0.2.0) for Microsoft Windows(32 bit)
  - Micro Focus Visual COBOL 2.3J
  - Microsoft Visual Studio Professional 2015 Version 14.0.24720.00

  ハードウェア

機種：	Dell OPTIPLEX 7010
CPU：	Intel Core i7-3770 CPU 3.40GHz
Memory：	4.00 Gbyte (ゲスト OS に割り当てたサイズ)

【第２回検証】

- **Database サーバー、Visual COBOL 開発サーバー**

  ソフトウェア

  - Windows Server 2012R2
  - Oracle Database 12c Release 1(12.1.0.2.0) for Microsoft Windows(x64)
  - Micro Focus Visual COBOL 2.3J Update 1
  - Microsoft Visual Studio Professional 2015 Version 14.0.25123.00
  -

  ハードウェア

機種：	n/a(public クラウドを利用)
CPU：	Intel Xeon CPU E5-2670 v2 2.50GHz
Memory：	3.75 Gbyte

3. テスト内容

Oracle Database Examples には Pro*COBOL 用のサンプルプログラム及び Micro Focus 製品向けのコンパイル・ビルドスクリプトが含まれます。本検証ではこのサンプルプログラムのうち SAMPLE4.pco として提供されるプログラムを利用しました。このプログラムには DDL 文、DML 文、DCL 文が含まれこれらの基本的な動作をテストできるようになっています。初めの Pro*COBOL を直接使った検証では、Oracle より提供される同プログラム及びビルドスクリプトを使用してコンパイル・ビルドしました。続く COBSQL を通じた検証においては、ビルドスクリプトの内容に合わせてコンパイル及びビルドの命令を構成しました。ビルドスクリプトは実行形式へビルドするよう記述されていますが、Micro Focus オリジナルの動的ロードモジュール形式 .gnt へのビルドも検証しています。

4. 結果

4.1 サンプルアプリケーションの取得

Database Server にインストールした Oracle Database 12c Release 1 Examples(12.1.0.2.0) for Microsoft Windows(x64) より該当のサンプルプログラム及びビルドスクリプトファイルを取得しました。

4.2 サンプルアプリケーションの確認

本検証で利用したサンプルアプリケーションには以下のような処理が含まれます。

① Oracle データベースに接続
② CREATE TABLE 文にてテスト用のテーブルを作成
③ サンプルスキーマに含まれる EMP テーブルよりデータを CURSOR – FETCH で取得
④ 取得したデータ及びプログラム中で加工したデータを INSERT 文にてテスト用のテーブルにデータ充填
⑤ 社員番号の入力を促すプロンプトを出力
⑥ 入力された社員番号をキーに SELECT INTO 文を発行
⑦ 取得したデータを表示
⑧ テスト用のテーブルを DROP
⑨ Oracle データベースとの接続を切断

4.3 サンプルアプリケーションの実行結果

Visual COBOL 2.3J 並びに Visual COBOL 2.3J Update 1 にてサンプルアプリケーションを正常に実行できることを確認しました。詳細は付録[1]の通りとなります。

5. テスト結果及び考察

Oracle が提供するプリコンパイラ Pro*COBOL が生成する COBOL プログラムを Visual COBOL 2.3J、Visual COBOL 2.3J Update 1 のいずれを用いても正常にコンパイルできることを確認しました。実行形式、動的ロードモジュール、形式問わずコンパイルしたアプリケーションが正常に動作することも確認しています。同アプリケーションが正常に処理されたことにより代表的な DDL 文、DML 文、DCL 文を Pro*COBOL と Visual COBOL の組み合わせで問題なく扱えることが検証できました。また、COBSQL を使ってシングルステップで同アプリケーションを Visual Studio IDE 上でコンパイルできることも確認できました。つまり、実際にプログラマがメンテナンスをする埋め込み SQL 文が入ったままのソースを直接 IDE 上で管理できることが確認できています。これにより、Visual COBOL が COBOL 開発用に IDE に作りこんだ開発補助機能を駆使して、Oracle データベースと連携する COBOL アプリケーション開発作業の効率性を著しく向上させることが可能となります。

以上

[1] 付録では第 1 回目の検証（Visual COBOL 2.3J を用いた検証）の結果を記しています。

付録１. Windows – Pro*COBOL

1) プログラムの先頭行にてコンパイラオプションを上書き

```
C:¥work¥procob¥wp¥cmd>more sample4.pco
      $SET CURRENCY-SIGN(36) <
                                                                追加したコンパイラ指令

      *****************************************************************
      * Sample Program 4:  Datatype Equivalencing                     *
      *                                                               *
      * This program simulates the storage and retrieval of bitmap    *
      * images into table IMAGE, which is created in the SCOTT        *
                                    :
C:¥work¥procob¥wp¥cmd>
```

日本語ロケール配下でコンパイルする場合、CURRECNY-SIGN コンパイラオプションにはデフォルトでは「¥」(92、X'5C')が指定されます。Oracle から提供されるサンプルプログラムでは通貨記号に「$」(36、X'24')を使用しているため、コンパイラに「$」を通貨記号として認識させるべくオプションを上書きます。

2) プログラム中の接続処理部分を検証環境に合わせて書き換え

編集前：

```
C:¥work¥procob¥wp¥cmd>more sample4.pco
                                    :
       01  USERNAME          PIC X(10) VARYING.
       01  PASSWD            PIC X(10) VARYING.
                                    :
           MOVE "scott" TO USERNAME-ARR.
           MOVE 5 TO USERNAME-LEN.
           MOVE "tiger" TO PASSWD-ARR.
           MOVE 5 TO PASSWD-LEN.
           EXEC SQL
               CONNECT :USERNAME IDENTIFIED BY :PASSWD
           END-EXEC.
           DISPLAY " ".
           DISPLAY "CONNECTED TO ORACLE AS USER:  ", USERNAME-ARR.
           DISPLAY " ".
                                    :
C:¥work¥procob¥wp¥cmd>
```

編集後

```
C:¥work¥procob¥wp¥cmd>more sample4.pco
                                    :
       01  CONNSTR           PIC X(25) VARYING.
                                    :
           MOVE "scott/tiger@cloudorcl" TO CONNSTR-ARR.
           MOVE 21 TO CONNSTR-LEN.
           EXEC SQL
               CONNECT :CONNSTR
           END-EXEC.
           DISPLAY " ".
           DISPLAY "CONNECTED TO ORACLE AS USER:  ", CONNSTR-ARR(1:5).
           DISPLAY " ".
                                    :
C:¥work¥procob¥wp¥cmd>
```

3) Pro*COBOL のサンプル中含まれるバッチファイル makeit.bat 中の 97 行目及び 105 行目をクライアント環境に合わせたパスに変更

```
C:¥work¥procob¥wp¥cmd>type makeit.bat
                                    :
cbllink %pcofile% /M%pcofile% %ORACLE_HOME%¥precomp¥lib¥%SQLLIB_lib%
                                    :
cbllink %pcofile% /M%pcofile% %ORACLE_HOME%¥precomp¥lib¥%SQLLIB_lib%
                                    :
C:¥work¥procob¥wp¥cmd>
```

4) 3) で編集した makeit.bat 使って、サンプルプログラムをプリコンパイル及びコンパイル

```
C:¥work¥procob¥wp¥cmd>makeit SAMPLE4

C:¥work¥procob¥wp¥cmd>procob iname=SAMPLE4.pco ireclen=132

Pro*COBOL: Release 12.1.0.2.0 - Production on 月 10月 19 12:01:20 2015

Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights reserved.

システムのデフォルト・オプション値:
C:¥app¥client¥tarot¥product¥12.1.0¥client_1¥precomp¥admin¥pcbcfg.cfg


C:¥work¥procob¥wp¥cmd>echo Compiling SAMPLE4......
Compiling SAMPLE4......

C:¥work¥procob¥wp¥cmd>cobol SAMPLE4 /anim /litlink /ibmcomp /NESTCALL makesyn
"COMP-5" = "COMP";
Micro Focus COBOL
Version 2.3.00341 Copyright (C) Micro Focus 1984-2015. All rights reserved.
* 無視  - NESTCALL
* チェック終了：エラーはありません- コード生成を開始します
* Generating SAMPLE4
* Data:       10096     Code:        6272     Literals:        2452

C:¥work¥procob¥wp¥cmd>cbllink SAMPLE4 /MSAMPLE4
C:¥app¥client¥tarot¥product¥12.1.0¥client_1¥precomp¥lib¥orasql12.lib
Micro Focus COBOL - CBLLINK utility
Version 2.3.0.84 Copyright (C) Micro Focus 1984-2015. All rights reserved.

Microsoft (R) Incremental Linker Version 11.00.61030.0
Copyright (C) Microsoft Corporation.  All rights reserved.

SAMPLE4.obj
cbllds00001188.obj
   Creating library SAMPLE4.lib and object SAMPLE4.exp
Microsoft (R) Manifest Tool version 6.2.9200.20789
Copyright (c) Microsoft Corporation 2012.
All rights reserved.

C:¥work¥procob¥wp¥cmd>
```

※1

※2

※3

※1: Pro*COBOL によるプリコンパイル。sample4.pco から sample4.cbl を生成。

※2: Visual COBOL のコマンドを使って sample4.cbl をコンパイル。

※3: Visual COBOL のコマンドを使って生成されたオブジェクトをビルド。

5) s5 スイッチの有効化

```
C:¥work¥procob¥wp¥cmd>set COBSW=/s5

C:¥work¥procob¥wp¥cmd>
```

6) プログラムを実行

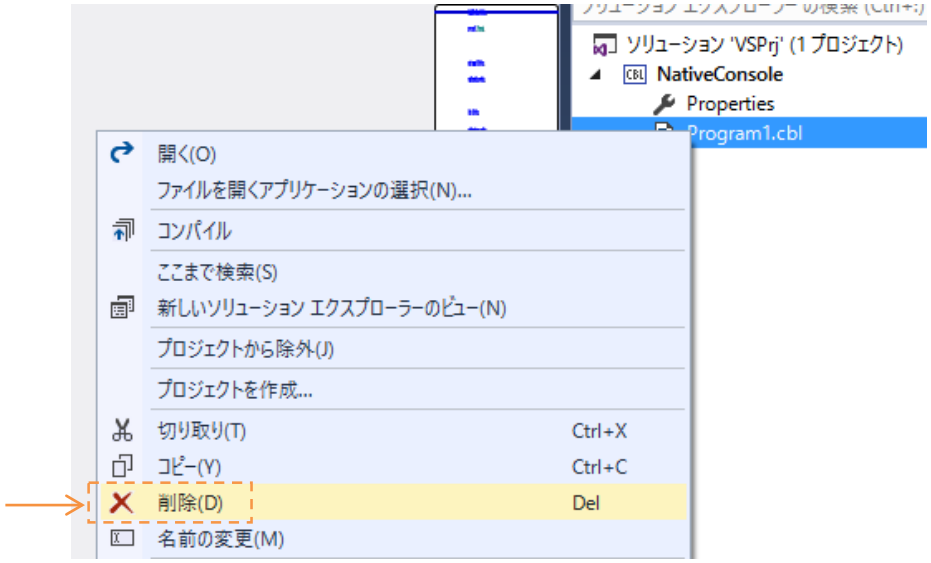
```
C:¥work¥procob¥wp¥cmd>SAMPLE4.exe

CONNECTED TO ORACLE AS USER:  scott

OK TO DROP THE IMAGE TABLE? (Y/N)  Y

TABLE IMAGE DOES NOT EXIST - CREATING NEW TABLE.

INSERTING BITMAPS INTO IMAGE FOR ALL EMPLOYEES ...

EMPLOYEE SMITH     ..............................  IS DONE!
EMPLOYEE ALLEN     ..............................  IS DONE!
EMPLOYEE WARD      ..............................  IS DONE!
EMPLOYEE JONES     ..............................  IS DONE!
EMPLOYEE MARTIN    ..............................  IS DONE!
EMPLOYEE BLAKE     ..............................  IS DONE!
EMPLOYEE CLARK     ..............................  IS DONE!
EMPLOYEE KING      ..............................  IS DONE!
EMPLOYEE TURNER    ..............................  IS DONE!
EMPLOYEE JAMES     ..............................  IS DONE!
EMPLOYEE FORD      ..............................  IS DONE!
EMPLOYEE MILLER    ..............................  IS DONE!

DONE INSERTING BITMAPS.  NEXT, LET'S DISPLAY SOME.

ENTER EMPLOYEE NUMBER (0 TO QUIT):  7654

               ************************
               ************************
               ************************
               ************************
               ************************
               ************************
               ************************
               ************************
               ************************
               ************************

EMPLOYEE MARTIN     HAS SALARY $ 1250.00 AND COMMISSION $ 1400.00.

ENTER EMPLOYEE NUMBER (0 TO QUIT):  7900

               ************************
               ************************
               ************************
               ************************
```

```
                ************************
                ************************
                ************************
                ************************
                ************************
                ************************

EMPLOYEE JAMES      HAS SALARY $  950.00 AND NO COMMISSION.

ENTER EMPLOYEE NUMBER (0 TO QUIT):  0

HAVE A GOOD DAY.


C:¥work¥procob¥wp¥cmd>
```

➔ 全て正常に処理できていることが確認できます。

付録２. Windows – COBSQL(実行形式へのビルド)

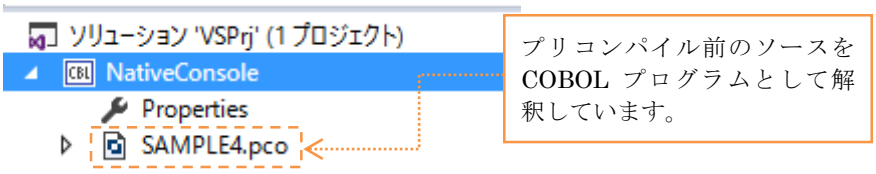
1) Visual Studio 2015 を起動
2) [ファイル]メニュー > [新規作成] > [プロジェクト] から
[テンプレート] > [COBOL プロジェクト] > [Native] を選択し、「コンソールアプリケーション」のプロジェクトテンプレートを選択してプロジェクトを作成
3) 自動で生成されるテンプレートプログラム Program1.cbl をソリューションエクスプローラにて右クリックの上、削除



4) 作成したプロジェクトを右クリックし、[追加] > [既存の項目] を選択し付録１で使用した sample4.pco をプロジェクトにインポート

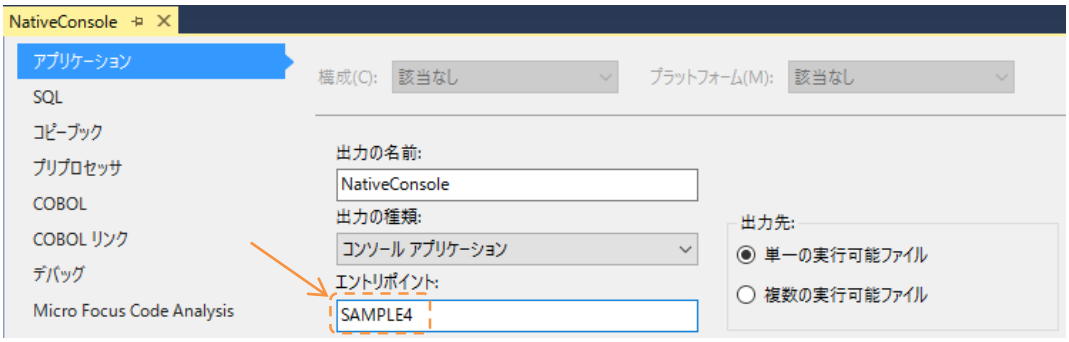
5)

インポート後の画面：

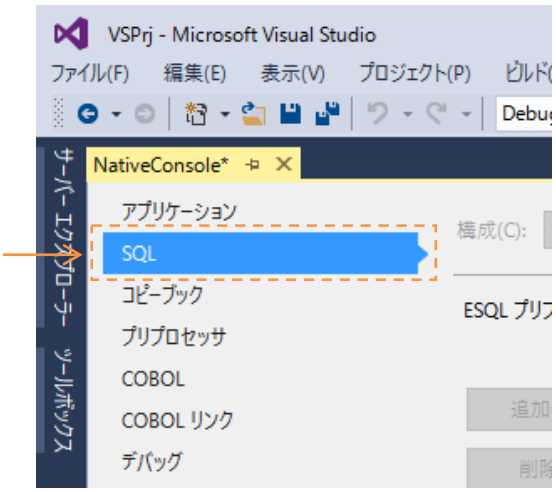


6) ソリューションエクスプローラにて［Properties］をダブルクリック

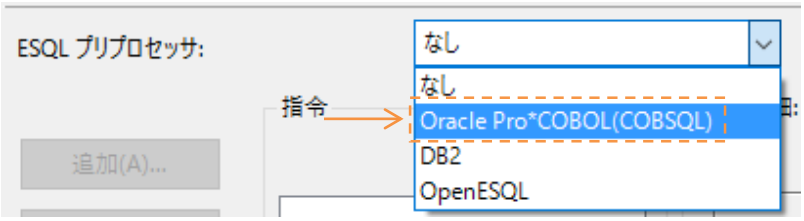
7) [アプリケーション]ページにてエントリポイントを指定



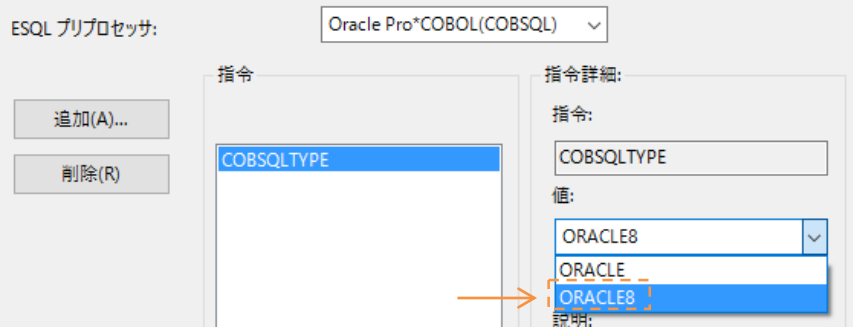
8) [SQL] ページを選択



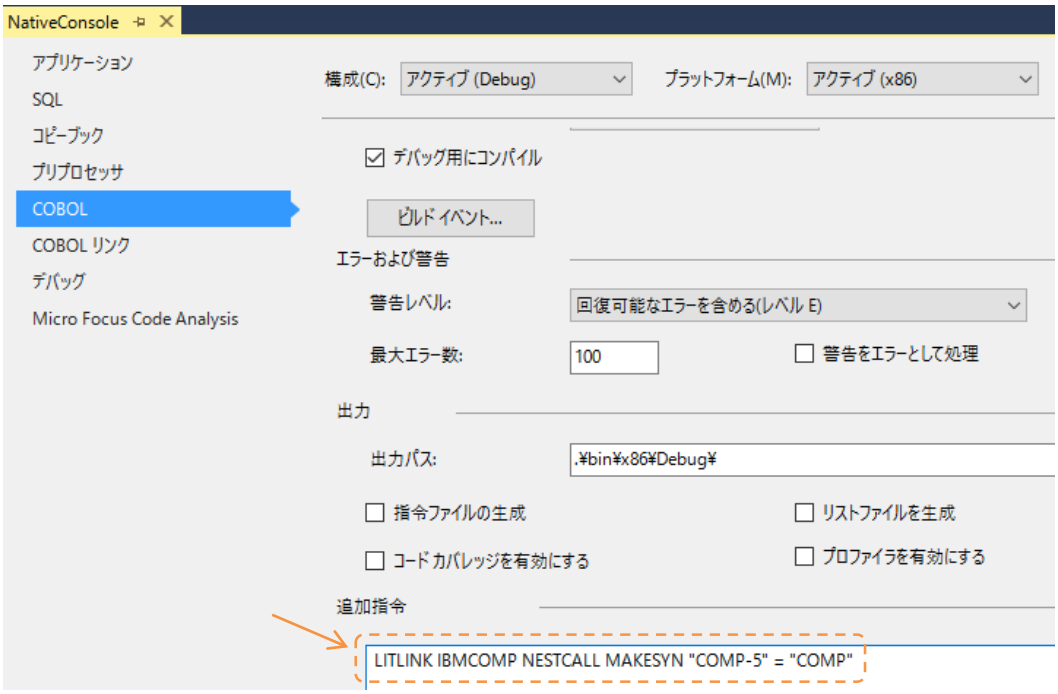
9) [ESQL プリプロセッサ] 欄にて「Oracle Pro*COBOL(COBSQL)」を選択



10) COBSQL オプションの選択画面では［COBSQLTYPE］に「Oracle8」[2]を指定

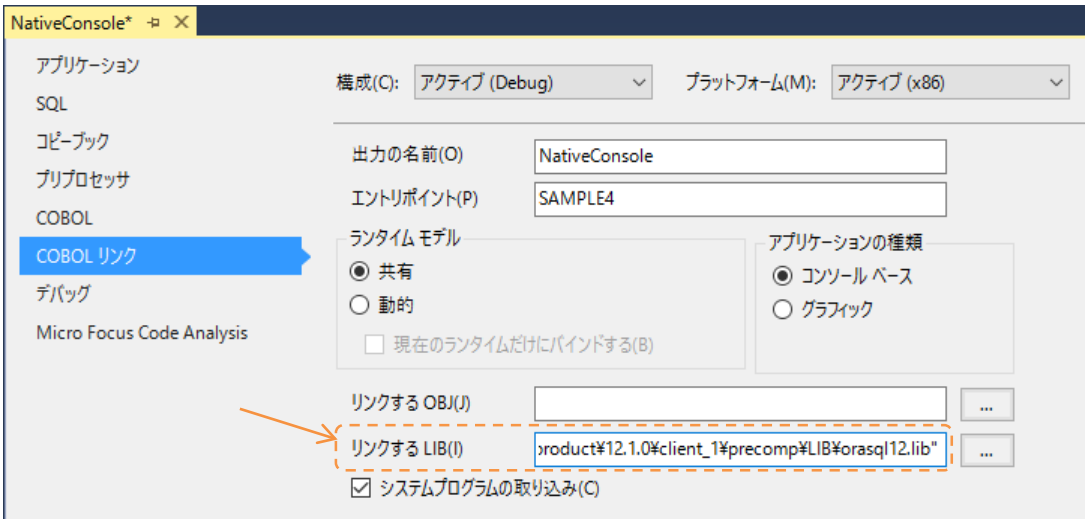


11) [COBOL]ページにて Oracle が提供するバッチファイル makeit.bat に記載されたコンパイラオプション及び COBSQL を有効にする旨のオプションを［追加指令］欄に指定[3] [4]



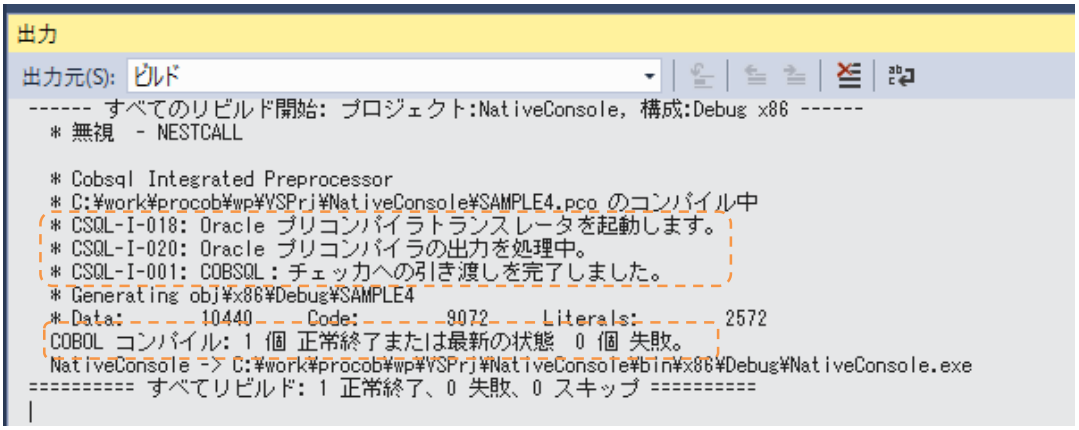
[2] Oracle 8 以降の Oracle とやりとりする際に指定します。
[3] ビルド設定欄で表示されているように ANIM 指令に相当する命令は同ページ中の設定項目「デバッグ用にコンパイル」で既に有効となっているため、ここでは割愛しています。
[4] NESTCALL 指令は Server Express のあるバージョンで廃止された指令であり、ここでの指定の有無は挙動に影響を与えません。

12) [COBOL リンク] ページにおける [リンクする LIB] 欄にて makeit.bat に倣い Oracle が提供するライブラリ **orasql12.lib** をリンクイン指定[5]



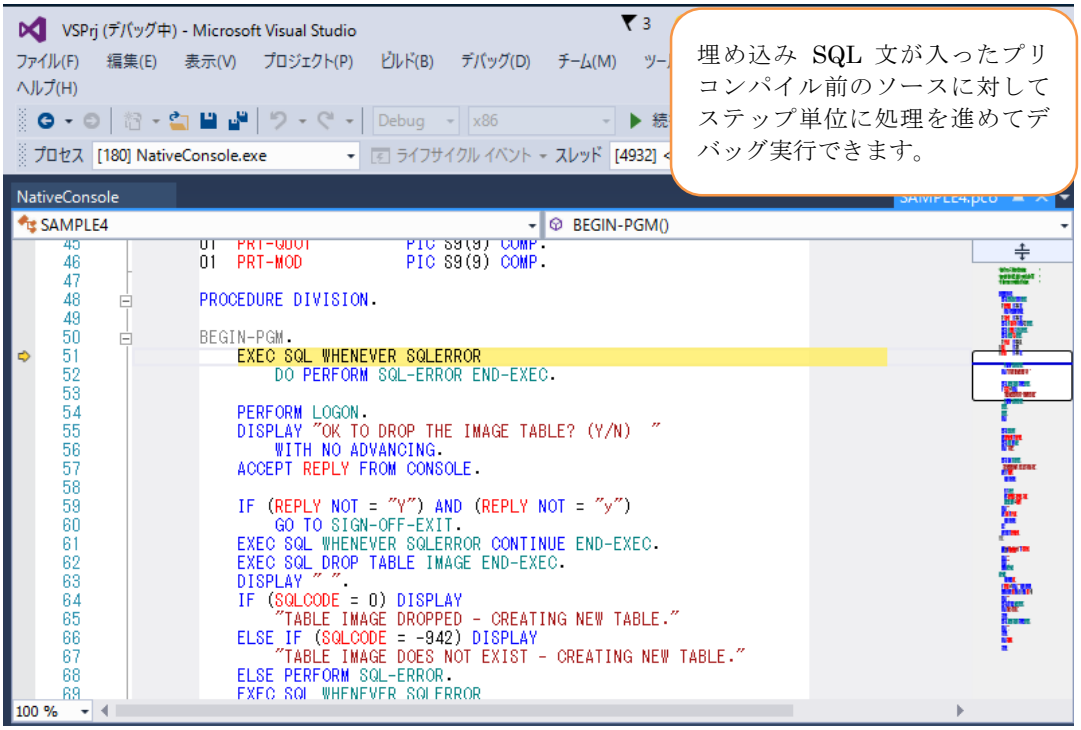
13) [ビルド]メニュー > [ソリューションのリビルド] を選択し、プログラムをビルド

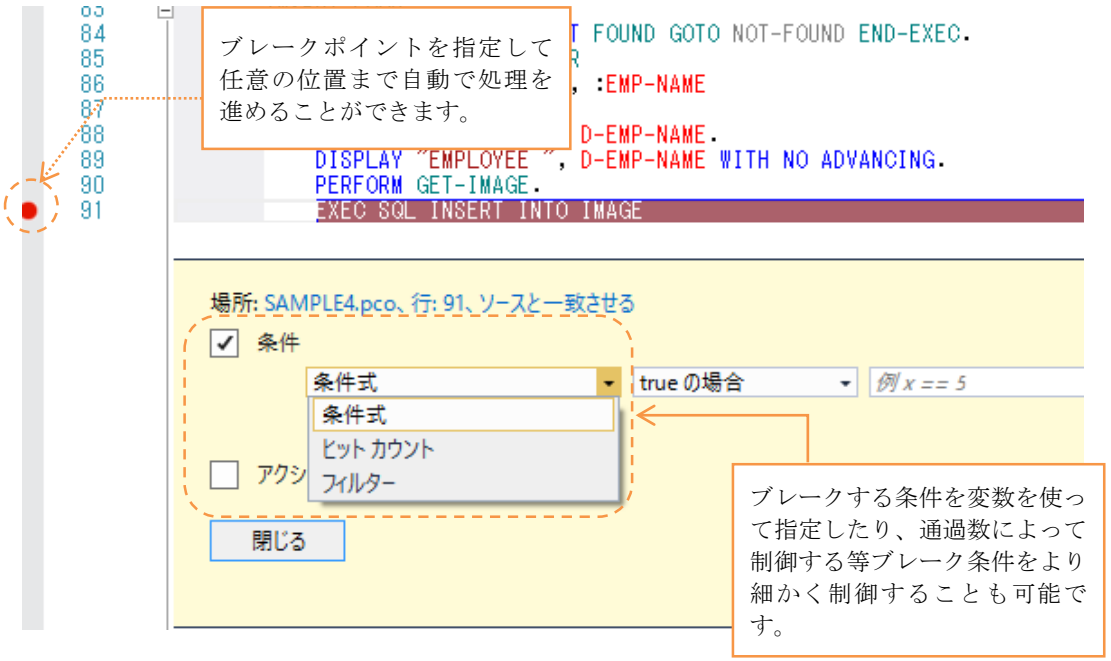
➔ 下図のように [出力] ビューにて正常にビルドされたこと並びに COBSQL により内部的に Pro*COBOL が利用されていることが確認できます。

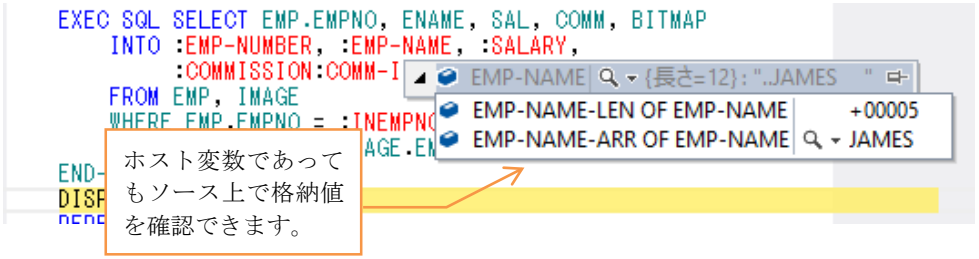


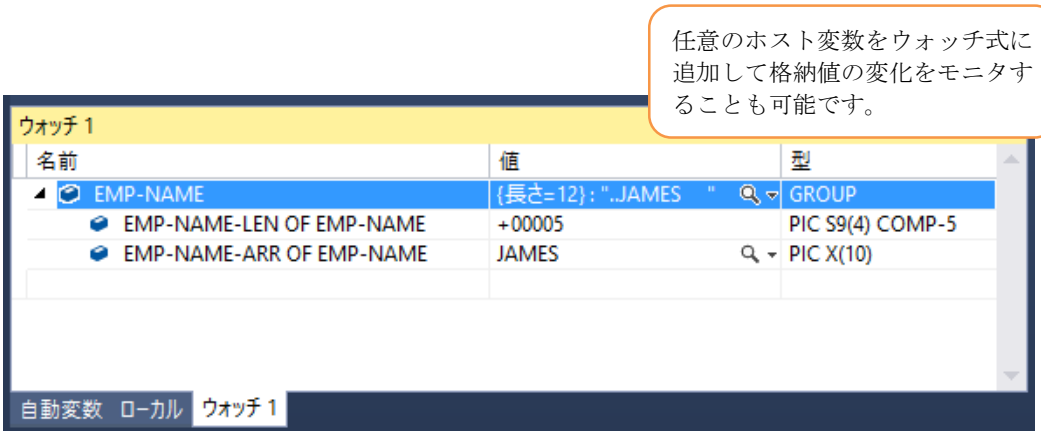
[5] 付録 1 でプリコンパイル展開されたソース SAMPLE4.cbl を確認すると SQLADR、SQLBEX 等をはじめとした Oracle API への CALL 文が多数追加されているのがわかります。実行形式やダイナミックリンクライブラリへビルドする際は本ステップで指定したライブラリを静的リンクしてこれらの API への参照を解決させます。

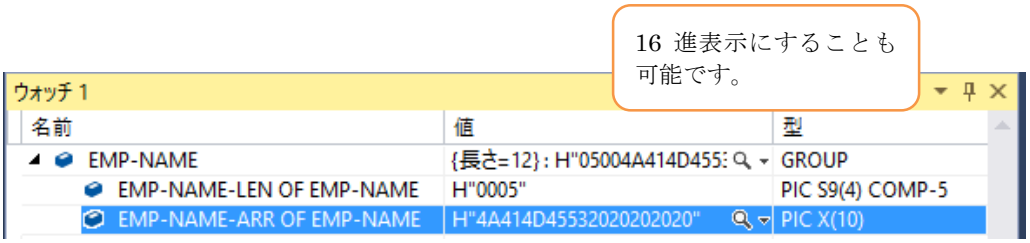
14) [デバッグ] メニュー > [ステップオーバー] を選択し、ステップ実行を開始

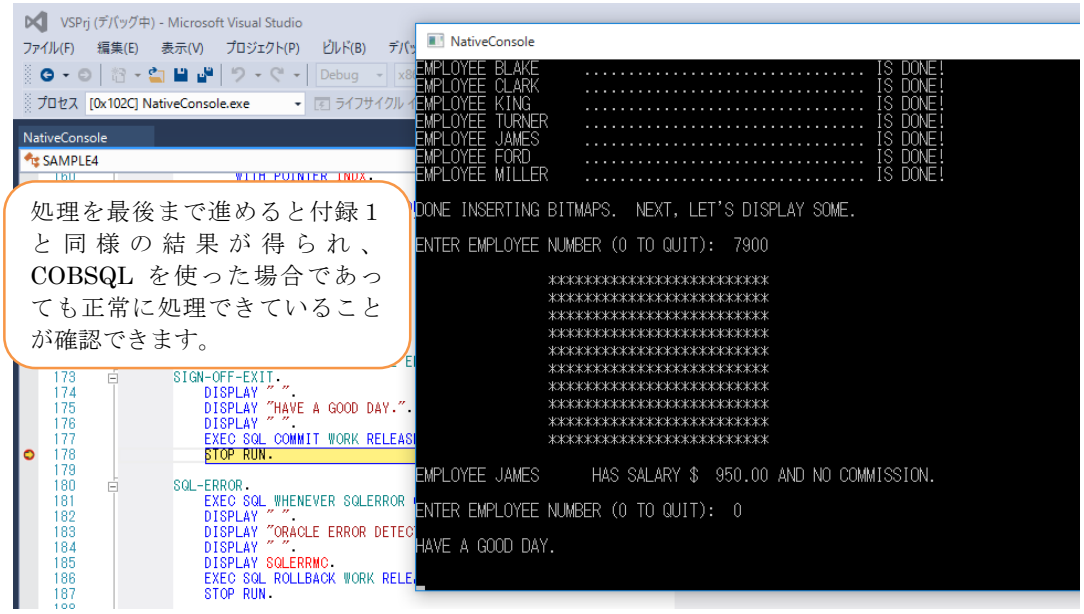





EXEC SQL SELECT EMP.EMPNO, ENAME, SAL, COMM, BITMAP
INTO :EMP-NUMBER, :EMP-NAME, :SALARY,
:COMMISSION:COMM-I
FROM EMP, IMAGE
EMP-NAME {長さ=12}: "JAMES "
EMP-NAME-LEN OF EMP-NAME +00005
EMP-NAME-ARR OF EMP-NAME JAMES
ホスト変数であってもソース上で格納値を確認できます。

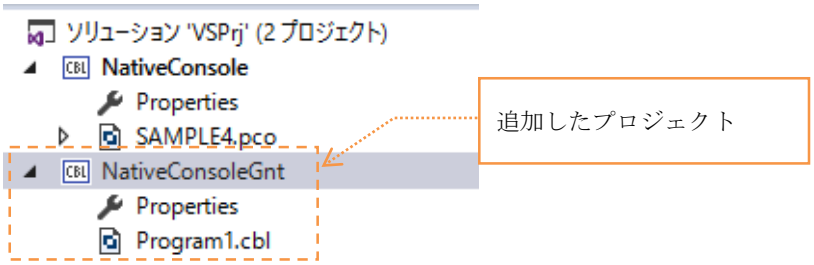

任意のホスト変数をウォッチ式に追加して格納値の変化をモニタすることも可能です。
ウォッチ 1
名前
値
型
EMP-NAME
{長さ=12}: "JAMES "
GROUP
EMP-NAME-LEN OF EMP-NAME
+00005
PIC S9(4) COMP-5
EMP-NAME-ARR OF EMP-NAME
JAMES
PIC X(10)
自動変数
ローカル
ウォッチ 1


16 進表示にすることも可能です。
ウォッチ 1
名前
値
型
EMP-NAME
{長さ=12}: H"05004A414D455
GROUP
EMP-NAME-LEN OF EMP-NAME
H"0005"
PIC S9(4) COMP-5
EMP-NAME-ARR OF EMP-NAME
H"4A414D45532020202020"
PIC X(10)

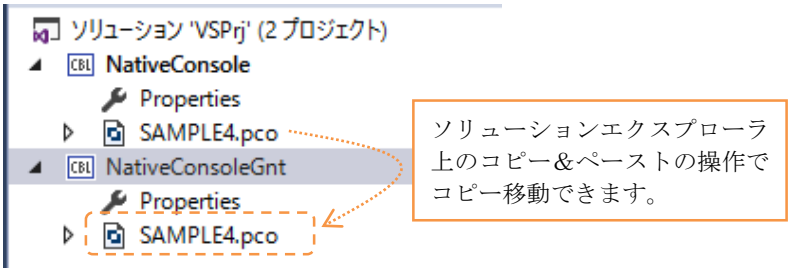

処理を最後まで進めると付録１と同様の結果が得られ、COBSQL を使った場合であっても正常に処理できていることが確認できます。
NativeConsole
EMPLOYEE BLAKE ..... IS DONE!
EMPLOYEE CLARK ..... IS DONE!
EMPLOYEE KING ..... IS DONE!
EMPLOYEE TURNER ..... IS DONE!
EMPLOYEE JAMES ..... IS DONE!
EMPLOYEE FORD ..... IS DONE!
EMPLOYEE MILLER ..... IS DONE!
DONE INSERTING BITMAPS. NEXT, LET'S DISPLAY SOME.
ENTER EMPLOYEE NUMBER (0 TO QUIT): 7900
EMPLOYEE JAMES HAS SALARY $ 950.00 AND NO COMMISSION.
ENTER EMPLOYEE NUMBER (0 TO QUIT): 0
HAVE A GOOD DAY.

付録３. Windows – COBSQL(動的ロードモジュールへのビルド)

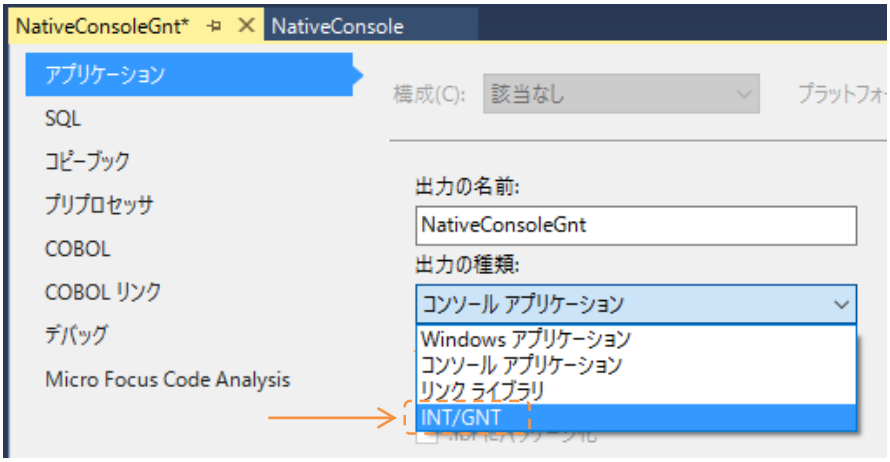
1) 付録２で用意した Visual Studio 2015 のソリューションを起動
2) ソリューションを右クリックし
   [追加] > [新しいプロジェクト]
   から Native のコンソールアプリケーションを追加



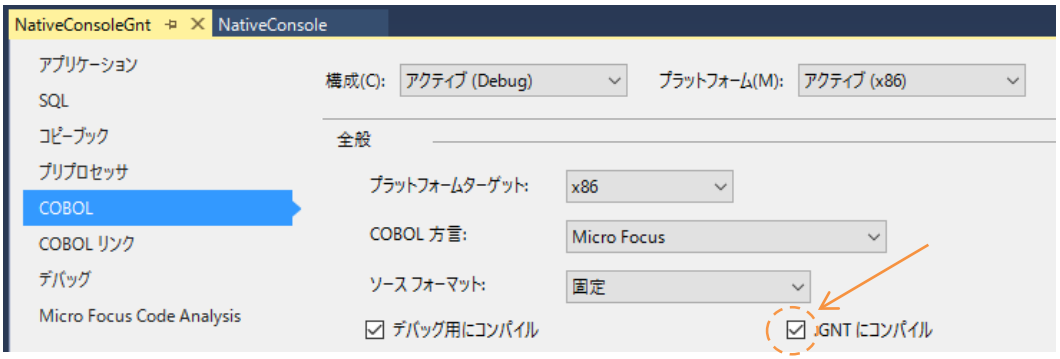
3) プロジェクトテンプレートに含まれる Program1.cbl を削除
4) SAMPLE4.pco を付録２で使用したプロジェクトからコピー



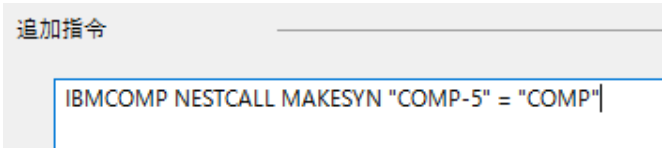
5) ソリューションエクスプローラにて [Properties] をダブルクリック
6) [アプリケーション]ページにおける [出力の種類] 欄にて [INT/GNT] を指定



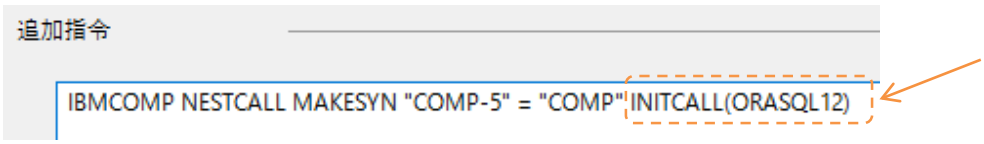
7) [SQL] ページにて [ESQL プリプロセッサ] 欄に「Oracle Pro*COBOL(COBSQL)」を指定
8) COBSQL 指令のうち [COBSQLTYPE] に「ORACLE8」を指定
9) [COBOL] ページにて GNT にコンパイルをチェック



10) [追加指令] 欄に付録２で追加したものと同じ指令を指定[6]



11) [追加指令] 欄に INITCALL(ORASQL12) を追加[7]



---

[6] 動的ロードモジュールはリンクを伴わないため、ここでは LITLINK 指令は除いています。

[7] 付録２で実行形式へビルドした際は SQLADR、SQLBEX 等の参照を orasql12.lib を静的リンクし解決させました。動的ロードの場合はライブラリを静的リンクするのではなく %ORACLE_HOME%¥bin¥orasql12.dll をプログラム実行前にロードして以降の Oracle API への参照を解決させるよう本ステップでは指定しています。

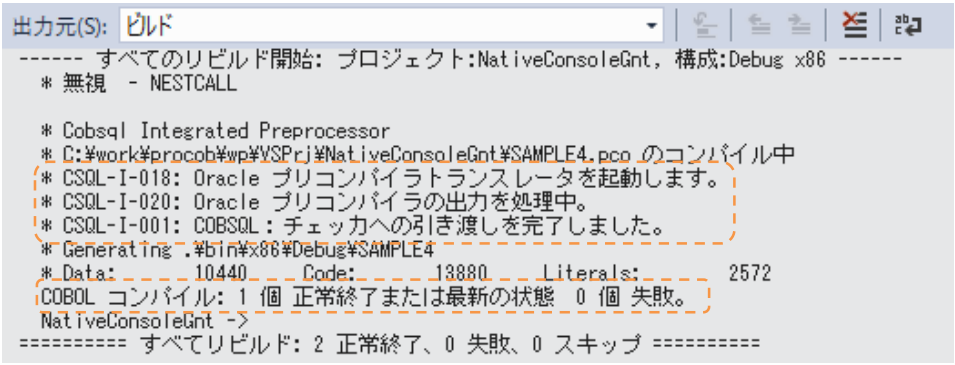
付録１でプリコンパイルされた SAMPLE4.cbl を見ると CONNECT 文の箇所で

```
      *     EXEC SQL
      *          CONNECT :CONNSTR
      *     END-EXEC.
            CALL "ORASQL8"
```

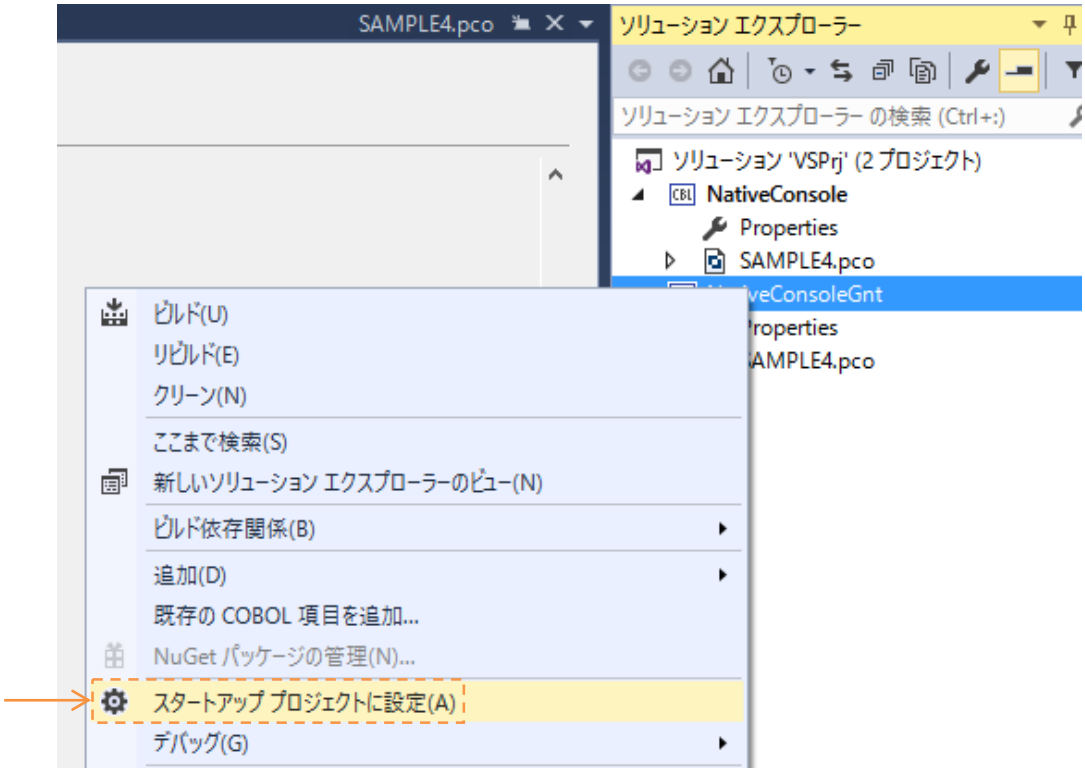
のように ORASQL8 を CALL しています。Oracle 12c クライアントをインストールすると orasql12.dll に加え、orasql8.dll もインストールされます。しかし、今回の検証ではこの orasql8.dll のロードだけでは Oracle API への参照を解決できず orasql12.dll を使用する必要があることがわかりました。そのため、実行時に最初に実行される SQL 文が CONNECT であることを確認できている場合は、orasql12.dll を orasql8.dll にリネームして使っても理論上は INITCALL(ORASQL12) と同じ挙動が得られます。

12) [ビルド]メニュー > [ソリューションのリビルド] を選択し、プログラムをビルド

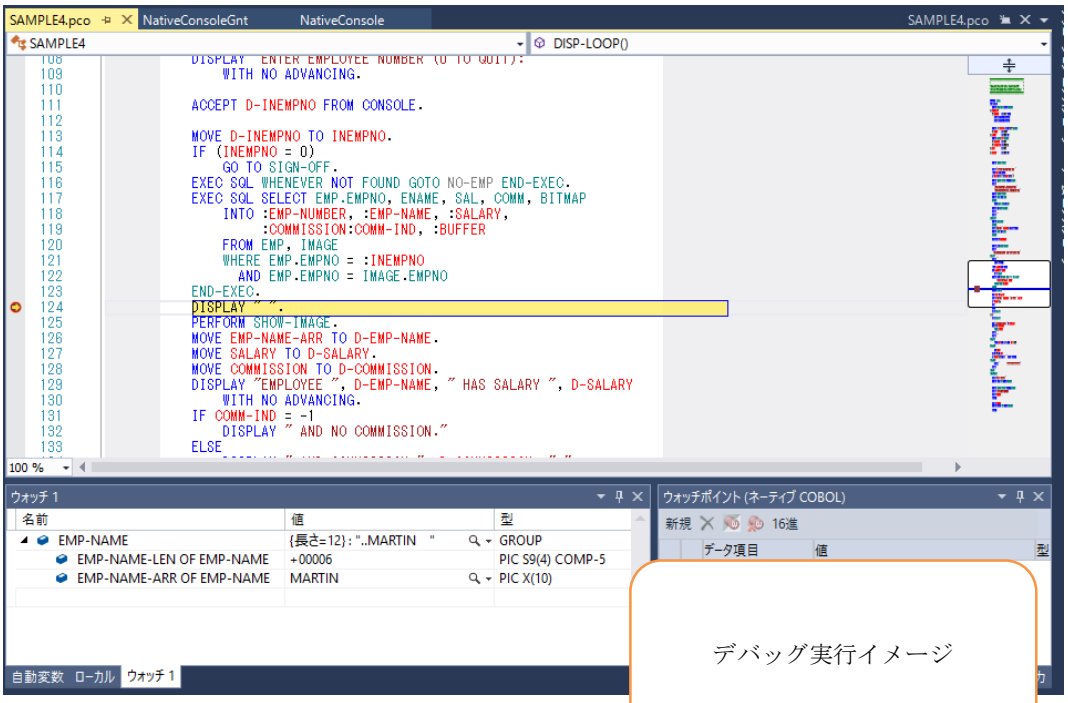
➔ 下図のように [出力] ビューにて正常にビルドされたこと並びに COBSQL により内部的に Pro*COBOL が利用されていることが確認できます。



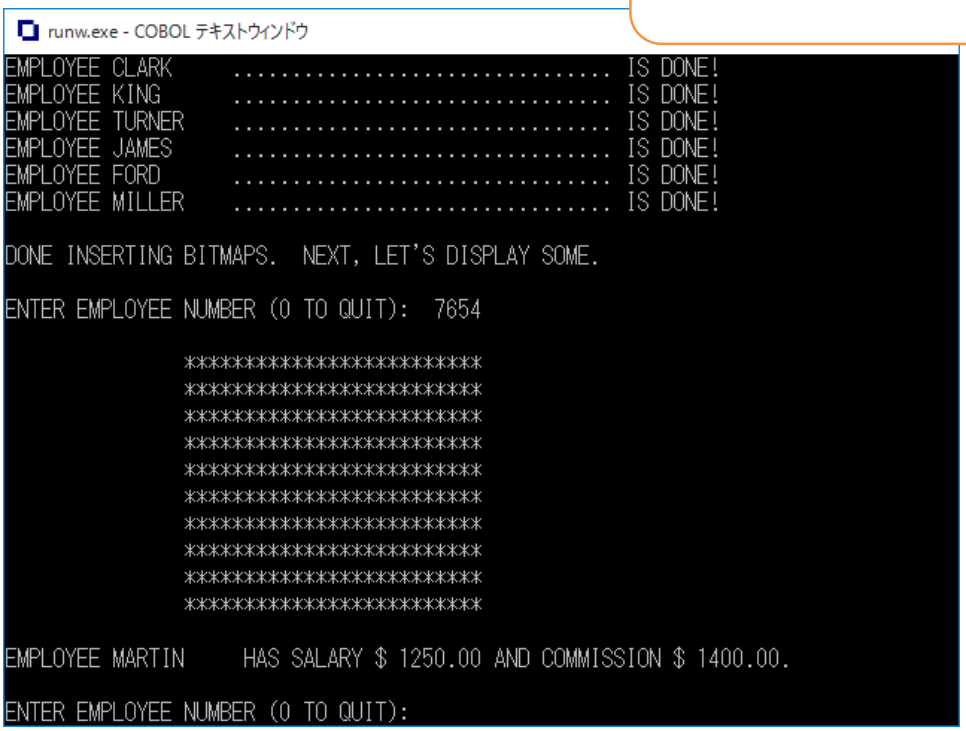
13) ソリューションエクスプローラにて動的ロードモジュール開発用に追加したプロジェクトを右クリックして [スタートアップ プロジェクトに設定] を選択



14) 付録２と同じ要領でデバッグ実行ができることを確認



デバッグ実行イメージ



以上